Webscraping Documentation

Richard Penman

Dec 06, 2020

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Background

For the last few years I have been specializing at web scraping and collected what I found useful into this library.

All code is pure Python and has been run across multiple Linux servers, Windows machines, as well as Google App Engine.

1.2 Install

Some options to install the webscraping package:

1. Checkout the repository: *hg clone https://code.google.com/p/webscraping/*
2. Download the zip: https://pypi.python.org/pypi/webscraping/
3. Install with pypi: *pip install webscraping*

The only dependency is python 2.5 or higher.

1.3 License

1.4 Contact

richard@webscraping.com

CHAPTER

TWO

EXAMPLES

2.1 Simple extraction

Except project title from the Google Code page:

```
from webscraping import download, xpath
D = download.Download()
# download and cache the Google Code webpage
html = D.get('http://code.google.com/p/webscraping')
# use xpath to extract the project title
project_title = xpath.get(html, '//div[@id="pname"]/a/span')
```

2.2 Blog scraper

Scrape all articles from a blog

```
import itertools
import urlparse
from webscraping import common, download, xpath

DOMAIN = ...
writer = common.UnicodeWriter('articles.csv')
writer.writerow(['Title', 'Num reads', 'URL'])
seen_urls = set() # track which articles URL's already seen, to prevent duplicates
D = download.Download()

# iterate each of the categories
for category_link in ('/developer/knowledge-base?page=%d', '/developer/articles?page=
↪%d'):
    # iterate the pages of a category
    for page in itertools.count():
        category_html = D.get(urlparse.urljoin(DOMAIN, category_link % page))
        article_links = xpath.search(category_html, '//div[@class="morelink"]/a/@href
↪')
        num_new_articles = 0
        for article_link in article_links:
            # scrape each article
            url = urlparse.urljoin(DOMAIN, article_link)
            if url not in seen_urls:
                num_new_articles += 1
                seen_urls.add(url)
```

(continues on next page)

(continued from previous page)

```
                html = D.get(url)
                title = xpath.get(html, '//div[@class="feed-header-wrap"]/h2')
                num_reads = xpath.get(html, '//li[@class="statistics_counter last"]/
↪span').replace(' reads', '')
                row = title, num_reads, url
                writer.writerow(row)
        if num_new_articles == 0:
            break # have found all articles for this category
```

2.3 Business directory threaded scraper

Scrape all businesses from this popular directory

```
import csv
import re
import string
from webscraping import common, download, xpath

DOMAIN = ...

class BusinessDirectory:
    def __init__(self, output_file='businesses.csv'):
        self.writer = common.UnicodeWriter(output_file)
        self.writer.writerow(['Name', 'Address'])

    def __call__(self, D, url, html):
        urls = []
        if url == DOMAIN:
            # crawl the index pages
            urls = [DOMAIN + '/atoz/%s.html' % letter for letter in string.uppercase␣
↪+ '#']
        elif re.search('/atoz/\w\.html', url):
            # crawl the categories
            urls = [DOMAIN + link for link in xpath.search(html, '//div[@id=
↪"partitionContainer"]//a/@href')]
        elif re.search('/atoz/\w/\d+\.html', url):
            # crawl the businesses
            urls = [DOMAIN + link for link in xpath.search(html, '//div[@id=
↪"listingsContainer"]//a/@href')]
        else:
            # scrape business details
            name = xpath.get(html, '//h1[@class="listingName"]')
            address = xpath.get(html, '//span[@class="listingAddressText"]')
            row = name, address
            self.writer.writerow(row)
        return urls

download.threaded_get(url=DOMAIN, proxies=proxies, cb=BusinessDirectory())
```

2.4 Daily deal threaded scraper

Scrape all deals from a popular daily deal website:

```
import re
import csv
import urlparse
from webscraping import common, download, xpath


DOMAIN = ...
writer = csv.writer(open('daily_deals.csv', 'w'))
writer.writerow(['Company', 'Address', 'Website', 'Email'])

def daily_deal(D, url, html):
    """This callback is called after each download
    """
    if url == DOMAIN:
        # first download - get all the city deal pages
        links = [link.replace('/deals/', '/all-deals/') for link in xpath.search(html,
↪ '//a[@class="jCityLink"]/@href')]
    elif '/all-deals/' in url:
        # city page downloaded - get all the deals
        links = re.findall('"dealPermaLink":"(.*?)"', html)
    else:
        # deal page downloaded - extract the details
        company = xpath.get(html, '//div[@class="merchantContact"]/h2')
        website = xpath.get(html, '//div[@class="merchantContact"]/a/@href')
        address = common.unescape(xpath.get(html, '//div[@class="merchantContact"]/
↪text()')).replace('Returns:', '').strip()
        if website:
            # crawl website for contact email
            email = '\n'.join(D.get_emails(website))
        else:
            email = None
        row = company, address, website, email
        # write deal details to CSV
        writer.writerow(row)
        links = []

    return [urlparse.urljoin(DOMAIN, link) for link in links]

# start the crawler
download.threaded_get(url=DOMAIN, proxy_file='proxies.txt', cb=daily_deal, num_
↪retries=1)
```

2.5 Navigate a website

Use webkit to navigate and interact with a website:

```
from webscraping import webkit
w = webkit.WebkitBrowser(gui=True)
# load webpage
w.get('http://duckduckgo.com')
# fill search textbox
w.fill('input[id=search_form_input_homepage]', 'webscraping')
# take screenshot of browser
w.screenshot('duckduckgo_search.jpg')
# click search button
w.click('input[id=search_button_homepage]')
# wait on results page
w.wait(10)
# take another screenshot
w.screenshot('duckduckgo_results.jpg')
```

CHAPTER

THREE

REFERENCE

3.1 `adt` Module

3.2 `alg` Module

3.3 `common` Module

3.4 `download` Module

3.5 `pdict` Module

3.6 `webkit` Module

3.7 `xpath` Module